MOTOROLA
Semiconductor Products Inc.

AN-880
Application Note

AN EVALUATION TOOL FOR THE MC68451 MMU

Prepared by
Hunter Scales
Microprocessor Applications Engineer
Austin, Texas

INTRODUCTION

In order to evaluate an LSI microprocessor peripheral, it is often necessary to design a prototype board which allows the programmer access to the peripheral. Occasionally, the peripheral is such that a different system can be used. In these cases, a "mezzanine" or "daughter" board can sometimes be used to add a new device to an already existing system, thus allowing evaluation of the new part without a complete system board. The board presented in this application note provides such an evaluation system for the MC68451 Memory Management Unit (MMU). To install this evaluation board, the MC68000 MPU should be removed from its socket on the host board of the user's system. The MPU should then be installed in the MPU socket of the "daughter" board, and the "daughter" board should then be plugged into the MPU socket on the host board. The address lines of the MPU now become the physical address bus from the MMU, and the MMU registers are now available to the programmer.

PRINCIPLES OF OPERATION

Refer to the circuit diagram in Figure 1 for the following discussion. The "daughter" board consists of six main blocks; the MPU header, the MC68000 MPU, the MC68451 MMU and its address/data demultiplexers, the MMU chip select circuitry, the physical strobe generation circuit, and the interrupt control circuit.

MPU Header

The block labeled MPU header represents the MC68000-compatible pinout on the underside of the "daughter" board. Its function is to bring the input from the host board to the MPU on the "daughter" board and then send the outputs from the "daughter" board to the host board. The data bus transports data to and from the MPU and to and from the MMU registers. The physical (translated) address bus is connected to the system address bus. The interrupt priority lines ($\overline{IPL0}$-$\overline{IPL2}$) are brought in from the host board for processing by the MPU. The clock (CLK) line brings in the system clock signal from the host board. The physical data strobes and physical address strobe are generated on the "daughter" board and sent to the host board via the header. All power for the "daughter" board is brought in through the power and ground pins on the header. This is sufficient for most systems, however, if the user's system is exceptionally noisy, external power supply lines may be required.

MC68000 MPU

The MC68000 device on the "daughter" board replaces the MC68000 on the host board. The MPU address bus then becomes the logical address bus and the upper 16 address lines are connected to the MMU to be translated into the upper order 16 physical address lines, while the lower seven lines are passed directly to the physical address bus.

MC68451 MMU and Support Circuits

The MMU provides the address translation mechanism for the system. The MMU has registers which must be read/written by the MPU. The registers are accessed by the chip select ($\overline{CS}$) signal on the MMU. The MMU chip select signal is generated by a decode circuit which is formed by three SN74LS266 EXCLUSIVE-NOR gates (U7, U8, and U2) and the 8-input NAND gate (U15).To activate U15 and provide a chip select to the MMU, the following requirements must be met: (1) address lines A6 and A7, physical address lines PA8 through PA11, and the physical address strobe (PAS) must all be high; and (2) each physical address line PA12 through PA23 must match its corresponding jumper in K1, K2, and K3. In this configuration, the MMU occupies the top 32 bytes in a block of 4096 memory locations. Jumpers K1, K2, and K3 allow for selection of any 4K block in the 16 megabyte addressing space of the MC68000.

As discussed above, the $\overline{CS}$ signal on the MMU is decoded from the physical address bus; therefore, the register addresses of the MMU are translated by the MMU itself. This requires that the MMU be assigned to a memory segment. Notice that assertion of the host board $\overline{RESET}$ line will also cause assertion of the MMU $\overline{CS}$ line. This initializes the MMU upon reset in such a way that it will pass the logical address through unmodified (transparently). This allows the system to operate without requiring that the MMU be specifically programmed beforehand.

The MC68451 has a multiplexed 16-bit port (PAD0-

PAD15) that serves as the bidirectional data bus and as the output port for the physical, translated address bus. The upper order 16 logical address lines are brought to the MMU from the MPU and translated according to the internal descriptors of the MMU. This translated address is output on the PAD0-PAD15 port and the $\overline{\text{HAD}}$ (hold address) signal is used to latch this address into the two SN74LS373 transparent latches (U11 and U12). If the MMU registers are then to be accessed by this address, the $\overline{\text{CS}}$ signal is asserted. This results in the PAD0-PAD15 port becoming the data bus for the MMU and the $\overline{\text{ED}}$ (enable data) signal is asserted to enable the two SN74LS245 bidirectional data buffers (U9 and U10). The direction of these buffers is controlled by the R/$\overline{\text{W}}$ line of the MPU.

Physical Strobe Generation

In addition to translating the physical address, the physical address strobe ($\overline{\text{PAS}}$) and the physical data strobes ($\overline{\text{PUDS}}$ and $\overline{\text{PLDS}}$) must also be generated. The $\overline{\text{PAS}}$ signal is generated by using the mapped address strobe ($\overline{\text{MAS}}$) signal from the MMU and the $\overline{\text{AS}}$ signal from the MPU. The $\overline{\text{MAS}}$ signal has three modes: the asynchronous mode, and two synchronous modes (S1 and S2). One of these modes is selected by jumper K4. In the asynchronous mode, the $\overline{\text{MAS}}$ signal requires an external delay line to form the $\overline{\text{PAS}}$ signal. This is done with U3C and U16. The delay is selectable by jumper K5 and should be equal to the amount of address setup time required by the target system (shown as 40 nanoseconds in Figure 1). In either of the synchronous modes, the delay is unnecessary and the delay input to U3C (pin 13) should be grounded. The $\overline{\text{PAS}}$ signal itself is then formed by passing through AND gate U6D and open collector inverter U4A.

The physical data strobes, physical upper data strobe ($\overline{\text{PUDS}}$) and physical lower data strobe ($\overline{\text{PLDS}}$), are formed by a combination of $\overline{\text{UDS}}$, $\overline{\text{LDS}}$, $\overline{\text{MAS}}$, write inhibit ($\overline{\text{WIN}}$) and the R/$\overline{\text{W}}$ line. The "logical" data strobes ($\overline{\text{UDS}}$ and $\overline{\text{LDS}}$) are asserted by the MPU at the start of a bus cycle. The $\overline{\text{MAS}}$ line is asserted by the MMU as soon as the translation is completed. This is sufficient in all cases except the case of the test and set (TAS) instruction, on the MC68000, which uses the read-modify-write cycle. It is possible that the write portion of the read-modify-write cycle could be attempted on a write protected segment. In this case, the write inhibit ($\overline{\text{WIN}}$) signal from the MMU prevents the cycle from altering data. This signal, together with the R/$\overline{\text{W}}$ line is used to inhibit the data strobes ($\overline{\text{PLDS}}$ and $\overline{\text{PUDS}}$) via U1B.

Interrupts

Since the MMU can cause interrupts to the MPU, interrupts must be handled in a special way. The IPL0-IPL2 inputs are brought into the "daughter" board through the MPU header. These inputs contain an inverted 3-bit encoded interrupt request level (level 0 is no request) which is fed into an SN74LS156 dual 2-to-4 decoder (U19) configured as a 3-to-8 decoder with open collecter outputs. The outputs of U19 are each connected to pullup resistors and to the inputs of an SN74LS148 8-to-3 priority encoder. These inputs are also available at jumper block K6. By jumpering the output of U4B to one of the U19 output lines, an $\overline{\text{IRQ}}$ interrupt from the MMU is inserted as one of the encoded interrupt inputs to the 8-to-3 priority encoder (U18). The MMU $\overline{\text{IRQ}}$ interrupt is then processed by the MC68000 MPU on the "daughter" board.

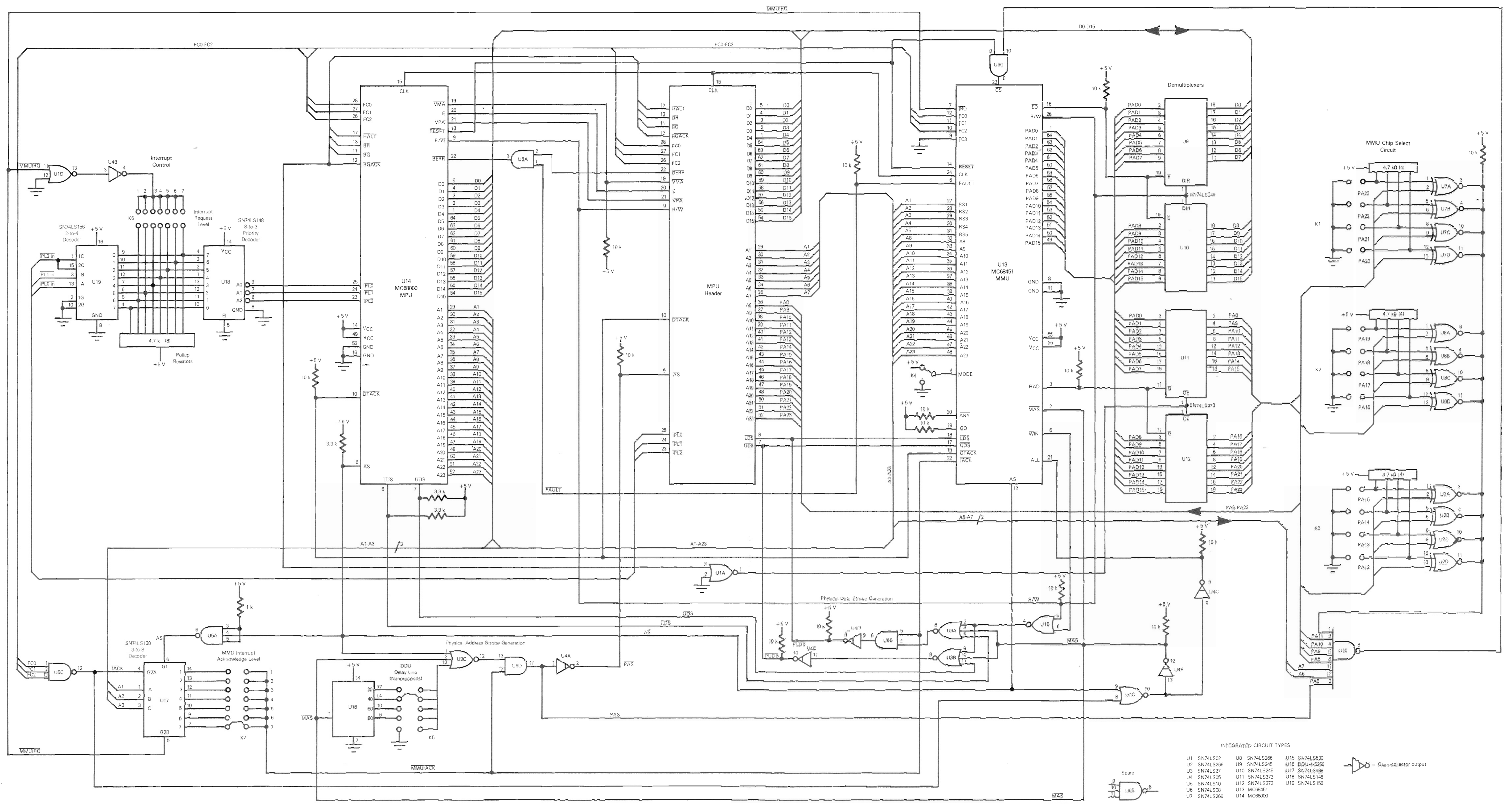
The interrupt acknowledge operation of the MMU causes it to place its interrupt vector on the low order data lines (D0-D7) for capture by the MPU during the $\overline{\text{IACK}}$ bus cycle. During this cycle, the MPU places the number of the acknowledged level on the lower three address lines (A1, A2, and A3) and drives A4-A23 high. These lines are brought to an SN74LS138 3-to-8 decoder (U17) and the acknowledge level is selected by jumper block K7 and sent to the $\overline{\text{IACK}}$ input on the MMU. This level must be the same as the interrupt request level of the MMU as selected by jumper K6. Notice also that the $\overline{\text{IACK}}$ signal, generated by U5C and U1C, is buffered by open collector inverters U4C and U4F before application as a low input to the $\overline{\text{MAS}}$ and ALL pins on the MMU. This has the effect of "fooling" the MMU into thinking that another MMU has translated the interrupt acknowledge "address". This allows the interrupt acknowledge cycle to be handled by the MMU without using a descriptor to map the $FFFFFX segment that the $\overline{\text{IACK}}$ bus cycle appears to be reading.

Since the MMU is physically isolated from the host board, it is desirable to prevent address and data strobes from being sent to the host board. This is accomplished by blocking $\overline{\text{PAS}}$ by U6D and the $\overline{\text{PLDS}}$ by U6B during the $\overline{\text{IACK}}$ bus cycle. This bus cycle is therefore not seen at all by the host system. This could also be accomplished with the MMU chip select circuit logic if the MMU register map should interfere with the existing system memory map.

TEST SOFTWARE

A listing which contains a short routine to test the "daughter" board is provided with this application note. The MMU is physically decoded at $FEFF00. First the MMU is set up such that descriptor 1 maps a 2K byte segment at logical location 0 to physical location 0. That is, the address is passed through from 0 to $3FF. Next, descriptor 2 is set up to map a 1K byte segment from $800 to $BFF transparently. Decriptor 3 is set up to map a 1K byte segment from physical address $C00 to logical address $4000 and the interrupts are enabled for this segment. The interrupt vector register (IVR) of the MMU is then loaded with $40 which is the vector at location $100 of the MPU. Now, when a location in the segment, starting at logical address $4000, is accessed, the MMU issues an interrupt request, and the interrupt handling logic can then be tested. Next, descriptor 4 is set up to map the ROM monitor to the host system by transparently mapping the 256K byte segment from $FC0000 to $FFFFFF. The address space table (AST) in the MMU is then set to use these descriptors. Descriptor 0 (which had been transparently mapping the entire address space since reset) is then disabled and the MMU begins mapping through descriptors 1,2,3, and 4.

After mapping as discussed above, a message is then sent to the terminal via the monitor I/O routines to the effect that the MMU has been successfully programmed. The monitor is then re-entered to allow testing the mapping of the segment, at address $4000, and the interrupts. This is done manually using the memory altering command of the monitor. If the interrupt occurs, the MMU is checked to see if the right descriptor caused the interrupt and appropriate messages are printed on the console.



INTEGRATED CIRCUIT TYPES

U1	SN74LS02	U8	SN74LS266	U15	SN74LS30
U2	SN74LS266	U9	SN74LS245	U16	DDU-4-5250
U3	SN74LS27	U10	SN74LS245	U17	SN74LS138
U4	SN74LS05	U11	SN74LS373	U18	SN74LS148
U5	SN74LS10	U12	SN74LS373	U19	SN74LS156
U6	SN74LS08	U13	MC68451		
U7	SN74LS266	U14	MC68000		

FIGURE 1 — MMU "Daughter" Board Schematic Diagram
Physical Address Strobe Generation Circuit

```
                      *
                      * THIS PROGRAM IS DESIGNED TO TEST THE INTERRUPT GENERATION CIRCUIT
                      * ON THE MMU DAUGHTER BOARD.
                      *
                      * MMU EQUATES
                      *          @          @
           00FEFF00   MMU        EQU        $FEFF00              MMU BASE ADDRESS
           00000000   AST        EQU        0                    ADDRESS SPACE TABLE OFFSET
           0000000A   AST5       EQU        $0A                  SUPERVISOR DATA ENTRY
           0000000C   AST6       EQU        $0C                  SUPERVISOR PROG ENTRY
           00000020   AC0        EQU        $20                  AC0 OFFSET
           00000026   AC6        EQU        $26                  AC6 OFFSET
           00000028   AC8        EQU        $28                  AC8 OFFSET
           0000003F   LD         EQU        $3F                  LOAC DESCRIPTOR OP
           00000029   DP         EQU        $29                  DESCRIPTOR POINTER
           0000002B   IVR        EQU        $2B                  INTERRUPT VECTOR REGISTER
           0000002D   GSR        EQU        $2D                  GLOBAL STATUS REG
           00000031   WSR        EQU        $31                  SEGMENT STATUS REG
           00000039   IDP        EQU        $39                  INTERRUPT DESCRIPTOR POINTER
                      *
                      *
                      *  MACSBUG RAM VECTOR TABLE
                      *
           00000000   VECTAB     EQU        0
                      *
           00000100   USERINT    EQU        $100                 USER INTERRUPT VECTOR AT $100
                      *
                      *
                      *  THE FOLLOWING IS THE ACTUAL DESCRIPTOR TABLE, 4 ENTRIES, 9 BYTES PER ENTRY
                      * IN THIS ORDER LBA(MSB), LBA(LSB), LAM(MSB), LAM(LSB), PBA(MSB),
                      * PBA(LSB), ASN, SSR, ASM.
                      *
                      *
0          00000000   DTAB       EQU        *                    START OF DESCRIPTOR TABLE
                      * DESCRIPTOR 1
0 00000000 0000                  DC.W       $0                   LOG BASE =0
0 00000002 FFF8                  DC.W       $FFF8                LAM OF 2K BYTES
0 00000004 0000                  DC.W       $0                   PHYS BASE OF $0 PASS THRU
0 00000006 01                    DC.B       01                   ADDRESS SPACE NUM
0 00000007 01                    DC.B       01                   SEG STAT=ENABLE SET
0 00000008 FF                    DC.B       $FF                  ASM IS ALL CARES

                      * DESCRIPTOR 2
0 0000000A 0008                  DC.W       $0008                LOG BASE=$800
0 0000000C FFFC                  DC.W       $FFFC                LAM OF 1K BYTES
0 0000000E 0008                  DC.W       $0008                LOG=PHYS
0 00000010 01                    DC.B       01                   ASN
0 00000011 01                    DC.B       01                   SSR
0 00000012 FF                    DC.B       $FF                  ASM

                      * DESCRIPTOR 3
0 00000014 0040                  DC.W       $0040                LBA =$4000
0 00000016 FFFC                  DC.W       $FFFC                LAM= 1K BYTES
0 00000018 000C                  DC.W       $000C                PBA= $C00
0 0000001A 01                    DC.B       01                   ASN
0 0000001B 11                    DC.B       $11                  SSR  SET I BIT
0 0000001C FF                    DC.B       $FF                  ASM

                      * DESCRIPTOR 4
0 0000001E FC00                  DC.W       $FC00                LBA= $FC0000-FFFFFF
0 00000020 FC00                  DC.W       $FC00                LAM= 256K BYTES
0 00000022 FC00                  DC.W       $FC00                PBA= FD0000
0 00000024 01                    DC.B       01                   ASN
0 00000025 01                    DC.B       01                   SSR
0 00000026 FF                    DC.B       $FF                  ASM

           0000C001              SECTION.S 1

1          00000000   MAIN       EQU        *
1 00000000 2E7C00000600          MOVE.L     #$600,A7             SET UP STACK POINTER
1 00000006 46FC2100              MOVE.W     #$2100,SR            SR=SUP,INT LEVEL 1
1 0000000A 41F900000000          LEA        DTAB,A0              A0 POINTS TO DESCRIPTOR TABLE
1 00000010 4280                  CLR.L      D0                   D0 WILL INDEX INTO DTAB
1 00000012 123C0001              MOVE.B     #1,D1                D1 WILL CONTAIN THE DESCRIPTOR NUMBER

1          00000016   NXTDESC    EQU        *
1 00000016 13C100FEFF29          MOVE.B     D1,MMU+DP            SET UP DESCRIPTOR POINTER
1 0000001C 615C                  BSR.S      LCDESC               LOAD THE DESCRIPTOR
1 0000001E 667A                  BNE.S      ERROR                CHECK FOR ERRORS
1 00000020 0600000A              ADD.B      #10,D0               TEN BYTES IN DESCRIPTOR
1 00000024 5201                  ADD.B      #1,D1                NEXT DESCRIPTOR
1 00000026 0C010005              CMP.B      #5,D1                HIT LAST YET ?
1 0000002A 66EA                  BNE.S      NXTDESC              DO NEXT
                      *
                      * NOW SET UP THE INTERRUPT VECTOR REGISTER WITH VECTOR #40 ($100)
                      *
1 0000002C 13FC00400CFE          MOVE.B     #$40,MMU+IVR         VECTOR NUMBER=$40= LOCATION $100
           FF2B

                      * ENABLE INTERRUPTS IN MMU, WRITE INT VECTOR ADDRESS INTO VECTOR TABLE

1 00000034 13FC000100FE          MOVE.B     #1,MMU+GSR           SET INT ENABLE BIT
           FF2D
1 0000003C 41F9C00000AC          LEA        INTPROG,A0           GET ADDR OF INT HANDLER
1 00000042 21C80100              MOVE.L     AC,VECTAB+USERINT    SET UP VECTOR

                      * NOW CHANGE THE SUPERVISOR ENTRIES OF AST TO TRANSLATE THROUGH
                      * DESCRIPTORS 1,2,3,AND 4

1          00000046   CHNGASN    EQU        *                    CHANGE THE AST
1 00000046 13FCC00100FE          MOVE.B     #1,MMU+AST5          CHANGE SUP DATA ENTRY TO 01
           FF0A
1 0000004E 13FCC00100FE          MOVE.B     #1,MMU+AST6          CHANGE SUP PROG ENTRY TO 01
           FF0C

                      * NOW DISABLE THE DESCRIPTOR 0
                      *
1 00000056 13FCC00000FE          MOVE.B     #0,MMU+DP            SET UP DES POINTER
           FF29
1 0000005E 13FCC00000FE          MOVE.B     #0,MMU+WSR           CLEAR THE SSR OF DES #0
           FF31

                      * DESCRIPTOR #0 IS NOW DISABLED
                      * REPORT SUCCESSFUL DESCRIPTOR LOAD TO USER
```

```

1 00000066 4BF9000000E6          LEA      MSG1,A5                SET FOR TRAP
1 0000006C 4DF9000000FE          LEA      EMSG1,A6
1 00000072 4E4F                  TRAP     #15
1 00000074 0002                  DC.W     2                      'DESCRIPTORS INITIALIZED'
1 00000076 4E4F         MACSBUG  TRAP     #15
1 00000078 0000                  DC.W     0                      GO TO MACSBUG

                        *  LDDESC LOAD DESCRIPTOR SUBROUTINE.  ENTER WITH:
                        *  D1 = DESCRIPTOR NUMBER TO BE LOADED
                        *  A0 = POINTER TO TABLE BASE WITH DESCRIPTOR PARAMETERS
                        *  D0 = OFFSET INTO TABLE TO SPECIFIC PARAMETERS FOR THIS DESCRIPTOR

1          0000007A     LDDESC   EQU      *                      LOAD DESCRIPTOR SUBROUTINE

1 0000007A 4CF000300000          MOVEM.L  0(A0,D0),D4-D5         MOVE LBA,LAM,PBA,ASN AND SSR TO D4 &D5
1 00000080 1C300008              MOVE.B   8(A0,D0),D6            PUT ASM IN D6

1 00000084 48F9003000FE          MOVEM.L  D4-D5,MMU+AC0          LOAD INTO ACCUM
           FF20
1 0000008C 13C600FEFF28          MOVE.B   D6,MMU+AC8             LOAD ASM INTO ACCUM
1 00000092 4A3900FEFF3F          TST.B    MMU+LD                 LOAD THE DESCRIPTOR AND CHECK STATUS
1 00000098 4E75                  RTS

                        * ERROR IF THERE IS AN ERROR IN LOADING A DESCRIPTOR (I.E., A COLLISION)
                        * REPORT TO USER

1          0000009A     ERROR    EQU      *
1 0000009A 4BF900000100          LEA      MSG2,A5
1 000000A0 4DF90000011A          LEA      EMSG2,A6
1 000000A6 4E4F                  TRAP     #15
1 000000A8 0002                  DC.W     2                      'ERROR IN DESCRIPTOR LOAD!'
1 000000AA 60CA                  BRA.S    MACSBUG

                        * INTPROG  INTERRRUPT ROUTINE TO REPORT AN INTERRUPT TO THE USER

1          000000AC     INTPROG  EQU      *                      INTERRUPTS FROM MMU COME HERE
1 000000AC 4BF90000011C          LEA      MSG3,A5                'INTERRUPT!!'
1 000000B2 4DF900000128          LEA      EMSG3,A6
1 000000B8 4E4F                  TRAP     #15
1 000000BA 0002                  DC.W     2

                        * NOW CHECK TO SEE IF MMU INTERRUPTED

1 000000BC 103900FEFF39          MOVE.B   MMU+IDP,D0             READ THE INTERRUPT DESCRIPTOR POINTER
1 000000C2 6B20                  BMI.S    NOTME                  NOT THE MMU
1 000000C4 0200001F              AND.B    #$1F,D0                MASK DESCRIPTOR NUMBER
1 000000C8 00000030              OR.B     #$30,D0                MAKE ASCII
1 000000CC 13C000000148          MOVE.B   D0,NUMBER              STORE TO OUTPUT
1 000000D2 4BF90000012A          LEA      MSG4,A5
1 000000D8 4DF90000014A          LEA      EMSG4,A6
1 000000DE 4E4F                  TRAP     #15
1 000000E0 0002                  DC.W     2                      'SEGMENT ACCESSSED, DESCRIPTOR # '
1 000000E2 6092                  BRA.S    MACSBUG
1 000000E4 4E73         NOTME    RTE

1 000000E6 444553435249 MSG1     DC.B     'DESCRIPTORS INITIALIZED'
1 000000FE 0000         EMSG1    DC.W     0

1 00000100 4552524F5220 MSG2     DC.B     'ERROR IN DESCRIPTOR LOAD!'
1 0000011A 0000         EMSG2    DC.W     0

1 0000011C 494E54455252 MSG3     DC.B     'INTERRUPT!!'
1 00000128 0000         EMSG3    DC.W     0

1 0000012A 5345474D454E MSG4     DC.B     'SEGMENT ACESSED, DESCRIPTOR #
1 00000148 01           NUMBER   DC.B     1                      PUT DESCRIPTOR NUMBER HERE
1 0000014A 0000         EMSG4    DC.W     0

                                 END
```